Bands and tiers

Common, Rare, Epic: `price_band` and `owners_tier` Names _____

Superset → **SQL → SQL Lab**, database `Dades classe`, schema `public`, table `games`, and your saved query `games_step2`. A category is a number made readable, like the rarity of an item. Pairs, one types and one writes; swap at Task 3. Run **one statement at a time**: select it, then **Run**.

Reference · the three rules of `CASE`

```
CASE WHEN condition_1 THEN 'label 1'
     WHEN condition_2 THEN 'label 2'
     ELSE 'label for the rest'
END AS new_column
```

Rule	What it means
Top to bottom	Each row stops at the first branch that is true.
`ELSE` takes the rest	Everything that matched nothing. Without `ELSE`, the rest becomes `NULL`.
`NULL` is never true	`NULL < 5` is unknown: a `NULL` falls through every comparison.

Task 1 · `price_band`

Five bands, prices in US dollars. Write the two missing branches (`5 to 15` and `15 to 30`). Then count.

```
SELECT CASE WHEN price IS NULL THEN 'Free'
            WHEN price < 5 THEN 'Under 5'
            -- your two branches here
            ELSE '30 or more' END AS price_band,
       COUNT(*) AS n
FROM games WHERE game IS NOT NULL
GROUP BY price_band ORDER BY MIN(price) NULLS FIRST;
```

Your two branches: _____

Free _____ Under 5 _____ 5 to 15 _____ 15 to 30 _____ 30 or more _____ Total _____

Why `ORDER BY MIN(price)` and not `ORDER BY price_band`? Try it. _____

Run `SELECT price, COUNT(*) FROM games GROUP BY price ORDER BY 2 DESC LIMIT 3;` The two most common prices: _____ and _____. In which band is each one? So why is the edge `< 5` and not `<= 5`?

_____

Task 2 · Break it twice

Copy your query and damage it, one change at a time. Write down what happens **and why**.

1 Move the `WHEN price < 30` branch to the very **top**, above all the others.

Bands now _____ The biggest: _____ with _____ games.

Where did the cheap games go? _____

2 Put it back. Now delete the `WHEN price IS NULL THEN 'Free'` line.

`30 or more` now has _____ games.

Where did the free games go, and which rule sent them there? _____

Task 3 · `owners_tier` on the clean query

Open `games_step2` (**SQL → Saved Queries**, click its name) and rename the block `step3`. Add two columns: your `price_band`, and this one. A column created in a `SELECT` cannot be reused in that same `SELECT`, so the `owners_min` expression is written out again:

```
CASE WHEN REGEXP_REPLACE(SPLIT_PART(owners, '..', 1), '[^0-9]', '', 'g')::bigint < 20000
       THEN 'Niche'
     WHEN REGEXP_REPLACE(SPLIT_PART(owners, '..', 1), '[^0-9]', '', 'g')::bigint < 1000000
       THEN 'Mid'
     ELSE 'Hit' END AS owners_tier,
```

Check with `SELECT owners_tier, COUNT(*) AS n FROM step3 GROUP BY owners_tier ORDER BY MIN(owners_min);`

Niche _____ Mid _____ Hit _____ Then save as `games_step3`: **Save**, change the name, **Save as new**.

Task 4 · Ask a question with both

Categories exist to be crossed. Keep the `WITH step3` block and change the final `SELECT`:

```
SELECT price_band, COUNT(*) AS games,
       SUM(CASE WHEN owners_tier = 'Hit' THEN 1 ELSE 0 END) AS hits,
       ROUND(100.0 * SUM(CASE WHEN owners_tier = 'Hit' THEN 1 ELSE 0 END) / COUNT(*), 1) AS hit_pct
FROM step3 GROUP BY price_band ORDER BY MIN(price) NULLS FIRST;
```

Highest share of hits: _____ with _____ % Lowest: _____ with _____ %

Does this prove that a higher price makes a hit? _____

Task 5 · Messy categories

Text categories get dirty too. How many publishers are there?

```
SELECT COUNT(DISTINCT publisher) AS as_written,
       COUNT(DISTINCT LOWER(publisher)) AS ignoring_case FROM games;
SELECT publisher, COUNT(*) FROM games
WHERE LOWER(publisher) IN ('big fish games', 'capcom') GROUP BY publisher ORDER BY 1;
```

`as_written` _____ `ignoring_case` _____ Spellings too many _____

The four spellings and their games: _____

`CASE WHEN LOWER(publisher) = 'big fish games' THEN 'Big Fish Games' ELSE publisher END` merges two of them.

What does its `ELSE` do with every other publisher? _____

Done when Every blank is filled · `games_step3` is in your saved queries. **Fast pair?** Bin one numeric column of your own open dataset into three or four bands.

Stuck?

- **`We can't seem to resolve the column "price_band"`**: a new column can only be used one level up, in the `SELECT` that reads `step3`.
- **`Error parsing near 'CASE'`**: the comma before it is missing.
- **Six bands instead of five**: a typo in one label. `'5 to 15 '` with a space is another category.
- **`games_step2` is not in Saved Queries**: ask the teacher for its SQL.

At home Design three or four bands for one numeric column of your own open dataset. Write the `CASE`, the counts, and one sentence defending where you put the edges.